Fourteenth USA/Europe Air Traffic Management Research and Development Seminar (ATM2021)

Protocol-Based Congestion Management for Advanced Air Mobility

Christopher Chin, Karthik Gopalakrishnan, Hamsa Balakrishnan
Massachusetts Institute of Technology
Cambridge, MA, USA
{chychin, karthikg, hamsa}@mit.edu

Maxim Egorov, Antony Evans
Airbus UTM
Sunnyvale, CA, USA
{maxim.egorov, tony.evans}@airbus-sv.com

***Abstract*—Advanced air mobility operations (e.g., air taxis and drone deliveries) are expected to significantly increase the demand for limited airspace resources. Two key characteristics of these operations are that flights will be scheduled with short lead times, and operators may be unable or reluctant, for reasons of privacy, to share flight intent information. Consequently, there is a need for congestion management algorithms that are efficient and fair in dynamic, reduced-information settings. In this paper, we address these challenges by designing a protocol that determines the "rules-of-the-road" for airspace access under these settings. The proposed protocol centers on the construction of priority queues to determine access to each congested volume of airspace. We leverage the concepts of backpressure and cycle detection to avoid gridlock and promote efficiency, and present several flight- and operator-level prioritization schemes. We evaluate the impacts of the prioritization schemes on system-wide and operator-level efficiency and fairness through extensive simulations of three scenarios: random flight patterns, cross-flows, and hub-based operations. In all scenarios, we find that backpressure prioritization yields the most efficient solution, and that accrued delay or dominant resource prioritization is the most fair depending on the user's choice of fairness metric.**

***Keywords*—Advanced air mobility, UAS traffic management, congestion control protocols, efficiency, fairness**

I. Introduction

We study the problem of congestion management for autonomous aircraft operations, with the goal of supporting advanced aerial mobility services including urban air mobility (UAM) and unmanned aircraft systems (UAS) [1]. The basic premise is that for safe operations, a minimum separation must be maintained between any two vehicles when in flight. To ensure that minimum separation requirements are not violated, it is common to define three-dimensional volumes of airspace, or *sectors*, with capacities that limit the number of vehicles that simultaneously occupy the sector [2]. The capacity of a sector is determined by several factors, such as the size and shape of the sector, separation requirements, coordination capabilities of vehicles flying in the sector, and typical traffic flow patterns within the sector. Our objective is to design a control algorithm that ensures sector capacity constraints are satisfied. This algorithm can be implemented as a traffic management service (e.g., through a UAS Service Supplier [3], a U-Space Service Provider [4], or a Provider of Services for UAM [5]) that helps facilitate safe and efficient airspace utilization.

This work was supported in part by A^3 (by Airbus) under contract number 40008574 and the NASA University Leadership Initiative (grant #80NSSC20M0163). This article solely reflects the opinions and conclusions of its authors and not any NASA entity.

Congestion and sector capacity constraints in air traffic management have traditionally been addressed by solving a centralized traffic flow optimization problem [6]. In this paradigm, the schedules and desired trajectories of all flights being planned are known in advance (as is the case with commercial aviation), and an optimization problem is solved to obtain revised schedules that ensure compliance with airport and sector capacities. Traffic management for advanced aerial mobility applications is likely not compatible with this approach for three main reasons. First, many of these applications are for on-demand services, with the demand for a flight materializing with little advance notice, making long-term aggregate planning impossible. Second, because of competitive factors, operators may not want to share the entire planned trajectory of their flights. Third, a fully centralized architecture is not expected to scale sufficiently to support the expected levels of demand. A federated architecture, in which multiple service suppliers support the traffic management of unmanned aircraft through coordination, has therefore been proposed [7]. Such coordination, however, requires the specification of a protocol for resource allocation. We therefore revisit the problem of airspace congestion management with a focus on developing protocols that enable large-scale, efficient and fair operations with dynamic demand and reduced information sharing. There are four properties that we desire of our proposed congestion management protocol:

1) *Efficiency:* The total delays experienced by the vehicles should be low.
2) *Reduced Information Sharing:* Vehicles should only need to share their intent for some subsequent pre-specified time-interval, and not for their entire flight.
3) *Scalability:* The protocol should be easy to implement at-scale in a federated architecture.
4) *Fairness:* The protocol should consider fairness across vehicles and operators, to incentivize user adoption.

A. Prior work

Congestion control protocols have been studied in several contexts, including communication networks [8], surface transportation [9], and air traffic management [10]. The solution approaches proposed in literature range from queue-length management protocols [11] to dynamic traffic routing, demand management [12], backpressure algorithms [13], [14], and optimal network flow management [6], [15]. A major focus of prior work has been on improving efficiency, i.e., increasing system throughput, or maximizing resource utilization. Furthermore, the decentralized nature of many of these approaches makes applications to large-scale systems (e.g., internet or road networks) tractable, while limiting the amount of information shared with a centralized agent.

Fair congestion control has been studied in the context of routing packets in communication networks [16]. The simplifying assumptions typically made, such as infinite buffers at congested resources, high traffic volumes that can be approximated as fluid flows, and the presence of only one congested resource in the path of a packet, are rarely satisfied in air traffic networks. As a result, fairness in air traffic management has generally only been evaluated either through first-come-first-served simulations [17], or in centralized settings with full information sharing [18], [19]. By contrast, we aim to explicitly incorporate fairness preferences into the design of a congestion management protocol.

The idea of distributed protocols for air traffic management is not a new one [20], [21]; researchers have proposed rules-of-the-road style protocols [22], Markov decision process models [23], and speed control algorithms [24]. Our contribution is the incorporation of fairness and reduced information sharing to this class of algorithms that has historically focused only on safety and efficiency.

B. *Contributions and findings*

We develop a congestion management protocol to ensure that airspace sector capacity constraints are not violated. The key contribution of our work is a framework that uses priority queues at sectors, using user-specified priority functions to determine which flights can access constrained sectors. This framework a) identifies cycles in a distributed manner to avoid gridlock, b) prioritizes the resolution of conflicts at sectors with a high "backpressure" to minimize downstream congestion or gridlocks, c) only requires sharing of the trajectory information for the next time step with neighboring sectors to improve scalability and reduce information sharing, and d) supports a wide variety of fairness objectives, for both individual flights and their operators.

Our main findings are as follows:

1) Intra-operator deconfliction (i.e., an operator deconflicts its own flights from each other before filing trajectory requests), results in lower system-wide efficiency and fairness. In other words, it is preferable to allow the protocols to perform any necessary deconfliction by considering all operators.
2) Prioritizing flights based on the backpressure metric maximizes efficiency (i.e., minimizes total delays).
3) Prioritizing flights based on the accrued delay metric maximizes one notion of system-wide fairness (i.e., minimizes the standard deviation of flight delays).
4) Dominant resource fairness (DRF) prioritization achieves fairness as defined by a metric that considers *excess* and *expected* operator delays. Furthermore, DRF results in lower delay for the operator whose flights impose less externality (defined as the expected delay when it is the sole operator) on the system.

C. *Paper outline*

We formulate the congestion management problem in Section II. We present the general framework of our protocol in Section III, and discuss specific prioritization schemes in Section IV. We evaluate the performance of our protocol using three traffic patterns in Section V and discuss notions of operator fairness. Finally, we conclude with a summary and directions for future work in Section VI.

II. Problem Setup

We consider a setting where there are no reroutes, and the only congestion management action that can be taken is when to allow a vehicle to enter a sector. Once a vehicle is in a sector, it cannot be forced to leave the sector. We consider a discrete-time setting, where each vehicle can only occupy one sector at any time. We use the following notation:

$\mathcal{T}$:	Set of time periods $\{1, \ldots, t, \ldots, T\}$
$\mathcal{S}$:	Set of sectors $\{1, \ldots, N_{sectors}\}$
$\mathcal{O}$:	Set of operators $\{1, \ldots, N_{operators}\}$
$\mathcal{V}$:	Set of vehicles $\{1, \ldots, N_{vehicle}\}$
$\mathcal{V}_o$:	Set of vehicles operated by $o \in \mathcal{O}$
$\mathcal{V}_a$:	Set of active vehicles, i.e., ready to depart or currently airborne
$C(s,t)$:	Capacity of sector $s \in \mathcal{S}$ at time $t \in \mathcal{T}$
$orig(i)$:	Origin of vehicle $i \in \mathcal{V}$
$dest(i)$:	Destination of vehicle $i \in \mathcal{V}$
$d(i)$:	Scheduled departure time of vehicle $i \in \mathcal{V}$
$a(i)$:	Scheduled arrival time of vehicle $i \in \mathcal{V}$
$x(i,t)$:	Sector for vehicle $i \in \mathcal{V}$ at time $t \in \mathcal{T}$
$\widehat{x}(i,t)$:	Intended sector at time $t+1$ for vehicle $i \in \mathcal{V}$ based on information at t
$\mathbf{x}(t)$:	Sectors for all vehicles i at time t
$\widehat{\mathbf{x}}(t)$:	Intended sectors for all vehicles i at time $t+1$ based on information at t
$\mathcal{G}$:	Vehicles that can proceed to their next sector
$\mathcal{H}$:	Vehicles that must hold in their current sector
$del(i)$:	Total delay assigned to vehicle i
$del(i,t)$:	Binary variable representing the delay assigned to vehicle i in time-step t
μ^{syst} :	Mean system delay, $\frac{1}{\lvert\mathcal{V}\rvert}\sum_i del(i)$
μ_o^{oper} :	Mean system delay for operator $o \in \mathcal{O}$, $\frac{1}{\lvert\mathcal{V}_i\rvert}\sum_{j\in\mathcal{V}_i} del(j)$
σ^{syst} :	Standard deviation of delays for all vehicles in the system, $\sqrt{\mathrm{Var}_{i\in\mathcal{V}}[del(i)]}$

We explore efficiency and fairness on flight- and operator-levels. At the flight-level, high system efficiency corresponds to low μ^{syst}, and high system fairness corresponds to low σ^{syst}. For operators, high efficiency corresponds to low μ_i^{oper}. Fairness across operators is more nuanced, as it is a function of the mean delays they experience (measured by μ_i^{oper}) and the baseline congestion contributed by their operations. This is discussed in further detail in Section V.D. Lastly, we define a safe congestion management protocol to be one that ensures $C(s,t)$ is not violated $\forall\, \mathbf{x}(t)$.

We are interested in developing a safe congestion management algorithm that maximizes efficiency and fairness for vehicles as well as operators. One approach would be to set up an optimization problem with $\mathbf{x}(t)$, $\widehat{\mathbf{x}}(t)$, and $C(s,t)$ at every time instant t to determine which vehicles can proceed (i.e., determine $del(i,t)$). However, this requires that all vehicles share their current location and intent with a central authority that will then solve the optimization problem. Instead, we seek to reduce information sharing by solving this problem with minimal information at each sector to determine which vehicles can access it in the next time-step.

We now describe our information exchange constraints in greater detail. We assume that each vehicle i conveys its intent to use sector $\widehat{x}(i,t)$ to that sector. Recall that if $\widehat{x}(i,t) = x(i,t)$, then by definition, the vehicle is allowed to stay in that sector. We further allow each sector i to communicate with all sectors j adjacent to i the identity of vehicles that want to access sector i. Crucially, sector i only shares the identity of these vehicles, but not the position. This is necessary for sectors to identify cycles (Section III.C). In addition, we allow sector i to signal a scalar value indicative of upstream congestion (i.e., the length of built-up queue) to its neighboring sector j (Section III.D). For example, sector i can convey to sector j that it has a queue of length 7 which is blocked by the vehicle wanting to proceed from i into j, but it does not reveal the location of these 7 vehicles. We assume that all sectors convey this information truthfully. An analysis of the incentive-compatibility of this mechanism is beyond the scope of this paper. We refer to this set of communication rules between sectors as the *information sharing constraints*.

In summary, *our goal is to develop a congestion management protocol that is safe, efficient, fair, and satisfies our information sharing constraints*.

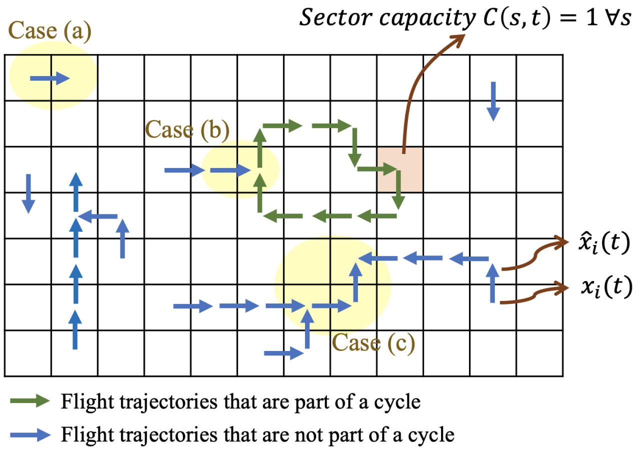
III. Congestion Management Protocol

In this section we develop the congestion management protocol, discuss a key feature that ensures that gridlocks are avoided, and present canonical approaches that serve as baseline prioritization schemes.

A. Setup

We make two simplifying assumptions. First, we set the capacity of each sector to one (although our approach generalizes to capacities greater than one). This can be realized in practice by defining a sector as a sufficiently small volume of airspace. Second, we assume that every vehicle either intends to a) stay in its current sector, or b) move to an adjacent sector. In practice, this means that the time discretization is small enough to capture the resource utilization of the vehicles.

Fig. 1 presents a simple visualization of the setup. Each grid cell is a sector with capacity 1. The tail and head of each arrow correspond to the present location $x(i,t)$ and intended location $\widehat{x}(i,t)$ of each vehicle. The congestion management algorithm needs to determine which vehicles can move along the direction of the arrow. In other words, if a vehicle $i \in \mathcal{V}$ has $del(i,t) = 1$, it means that it cannot travel to its intended sector and must try again in the next time period. In practice, vehicle i could absorb this delay with airborne holding, a speed change, or a path stretch, as long as it stays within $x(i,t)$. By contrast, a vehicle with $del(i,t) = 0$ is assigned no delay and can proceed to its next intended sector $\widehat{x}(i,t)$.



Case (a)
Sector capacity $C(s,t) = 1\ \forall s$
Case (b)
Case (c)
$\hat{x}_i(t)$
$x_i(t)$
Flight trajectories that are part of a cycle
Flight trajectories that are not part of a cycle

Figure 1: A simple example of the system state at time t, showing the current and intended sectors of all vehicles, with an example of a cycle in green.

We make a few observations about this problem setup. First, if there is a sector into which only one vehicle wants to enter, then the optimal—and trivial—solution would be to set $del(i,t) = 0$ for that vehicle. Case (a) in Fig. 1 shows an example of this. Second, in the scenario described in Case (b), we notice that vehicles with their trajectory intent marked in green form a "cycle". This means that either all of them are allowed to move, or none of them can. Furthermore, there is no feasible way in which any additional vehicle attempting to access the sectors occupied by the cycle can be allowed to do so while the cycle exists, because of capacity constraints. For example, the vehicles marked in blue that are incident on the green cycle cannot proceed while the green cycle exists. Third, Case (c) highlights a scenario in which there are multiple "connected" sectors where vehicles need to be deconflicted. A deconfliction decision at one of these sectors can have cascading effects on the decisions for the other sectors, so the order in which sectors are deconflicted is important.

B. Overview of the protocol

While being cognizant of the three observations on Fig. 1, we present the framework for our congestion management protocol, which is run at every time-step. We divide our protocol into six steps, with references to the appropriate line numbers in Algorithm 1.

Step 1: Initialization (Lines 1-3). We initialize two lists, a *hold* list $\mathcal{H}$ and a *go* list $\mathcal{G}$. Vehicles in $\mathcal{H}$ will be forced to hold and stay in the same sector in the next time-step, whereas vehicles in $\mathcal{G}$ will be allowed to proceed to their desired next sector. We update the list of *active* vehicles $\mathcal{V}_a$ by removing vehicles that have arrived at their destination and adding vehicles that are scheduled to take off.

Step 2: Identify and prioritize cycles (Lines 4-7). From Case (b) in Fig. 1, we know that cycles need to be identified and prioritized as soon as they appear. Until a cycle is cleared, it will block all sectors that it occupies. We identify vehicles in *cycles* ($\mathcal{V}_c$) and add them to $\mathcal{G}$. To make way for vehicles in $\mathcal{V}_c$, we add vehicles incident on cycles to $\mathcal{H}$ and force them to hold (i.e., their next sector is set to their current sector).

Step 3: Compute sector prioritization (Lines 8-9). Now that the cycles have been resolved (for this time-step at least), we need to decide the order in which to deconflict sectors. Case (c) in Fig. 1 shows an example of the dependencies between sectors. We calculate the backpressure at each sector and will deconflict the sector with the highest backpressure first. We will formalize the notion of backpressure in Section III.C, but it provides a measure of the queue build-up incident on a sector.

Step 4: Loop through sectors (Lines 10-12). Based on the order determined in step 3, we complete steps 4-6 for each sector. For the highest priority sector yet to be managed, we split the vehicles that want to use this sector in the next time-step into two categories: undecided vehicles ($\mathcal{V}_u$) and decided vehicles ($\mathcal{V}_d$). $\mathcal{V}_u$ contains vehicles that the sector is *undecided* on whether to allow them to enter the sector, and $\mathcal{V}_d$ contains vehicles for whom actions are *decided* (i.e., they are in either $\mathcal{G}$ or $\mathcal{H}$). Now, one of the two scenarios will occur:

Step 5a: Case of capacity exceeds demand (Lines 14-15). If the sector capacity is sufficiently high to allow all inbound traffic, then we add $\mathcal{V}_u$ to $\mathcal{G}$.

Step 5b: Case of demand exceeds capacity (Lines 17-21). If there is insufficient capacity to allow all vehicles, then we use one of several prioritization schemes to pick which vehicle gets to proceed. These prioritization schemes can be on a vehicle-level or operator-level and are described in detail in Section III.E and IV. These vehicles are removed from $\mathcal{V}_u$ and added to $\mathcal{V}_d$ and $\mathcal{G}$. We keep prioritizing vehicles until all capacity is used or there are no more vehicles in $\mathcal{V}_u$.

Step 6: Delay all unassigned vehicles (Lines 24-25). If capacity is fully used and there are still vehicles in $\mathcal{V}_u$, we add all $\mathcal{V}_u$ to $\mathcal{H}$ and force them to hold at their current sector.

Two main components of this algorithm are the FINDCYCLES and CALCULATEBACKPRESSURE functions, which we will describe next. A key requirement in designing these functions is that they satisfy the information-sharing constraints.

C. Computing cycles

The goal is for sectors to share limited information and identify vehicles incoming into it that are in cycles. We use an adapted Rocha-Thatte cycle detection distributed algorithm [25]. We have a finite directed graph $G := (\mathcal{S}, E)$ where the vertices are the set of sectors $\mathcal{S}$ and the edges are defined with tail $x(i,t)$ and head $\widehat{x}(i,t)$, $\forall i \in \mathcal{V}$. Under our assumptions, each sector is only aware of incoming and outgoing vehicles. We use rounds of "bulk synchronous message passing" to identify cycles. For each sector, we define three sets. The first is the set of incoming vehicles $\mathcal{V}_s^- = \{i \in \mathcal{V}_a \mid \widehat{x}(i,t) = s\}$. Next, we define a sector's in-neighbors as $\mathcal{N}_s^- = \{x(i,t),\ \forall i \in \mathcal{V}_s^-\}$. These are adjacent sectors that want to hand-off a vehicle to s. Similarly, we define a sector's out-neighbors as $\mathcal{N}_s^+ = \{\widehat{x}(i,t),\ \forall v \in \mathcal{V}_a \mid x(i,t) = s\}$.

Algorithm 1 PROTOCOL($\mathbf{x}, \widehat{\mathbf{x}}$, $\mathcal{V}_a$, $\mathcal{S}$, C)

```
1: $\mathcal{H} \leftarrow \{\}$,   $\mathcal{G} \leftarrow \{\}$
2: $\mathcal{V}_a \leftarrow \mathcal{V}_a \setminus (i \in \mathcal{V} \mid \mathbf{x}(i) = dest(i))$
3: $\mathcal{V}_a \leftarrow \mathcal{V}_a \cup (i \in \mathcal{V} \mid d(i) = t)$
4: $\mathcal{V}_c \leftarrow$FINDCYCLES$(\mathbf{x}, \widehat{\mathbf{x}})$
5: $\mathcal{G} \leftarrow \mathcal{G} \cup \mathcal{V}_c$
6: $\mathcal{H} \leftarrow \mathcal{H} \cup (i \in \mathcal{V} \mid \exists\, g \in \mathcal{G} \mid \widehat{x}(i,t) = \widehat{x}(g,t))$
7: $\widehat{x}(i,t) = x(i,t)\ \forall\ i \in \mathcal{H}$
8: $B \leftarrow$CALCULATEBACKPRESSURE$(\mathbf{x}, \widehat{\mathbf{x}}, \mathcal{V}_a, \mathcal{S})$
9: SORT $\mathcal{S}$ IN ORDER OF B
10: for $s \in \mathcal{S}$ do
11:     $\mathcal{V}_u \leftarrow i \in \mathcal{V} \mid \widehat{x}(i,t) = s$ and $\sim (i \in \mathcal{G}$ or $i \in \mathcal{H})$
12:     $\mathcal{V}_d \leftarrow i \in \mathcal{V} \mid \widehat{x}(i,t) = s$ and $(i \in \mathcal{G}$ or $i \in \mathcal{H})$
13:     if $C(s,t+1) > |\mathcal{V}_d|$ then
14:         if $|\mathcal{V}_u| \leq C(s,t+1) - |\mathcal{V}_d|$ then
15:             $\mathcal{G} \leftarrow \mathcal{G} \cup \mathcal{V}_u$
16:         else
17:             while $C(s,t+1) > |\mathcal{V}_d|$ do
18:                 $p \leftarrow$ PRIORITIZEVEHICLE$(\mathbf{x}, \widehat{\mathbf{x}}, \mathcal{V}_u)$
19:                 $\mathcal{G} \leftarrow \mathcal{G} \cup p$
20:                 $\mathcal{V}_u \leftarrow \mathcal{V}_u \setminus p$,   $\mathcal{V}_d \leftarrow \mathcal{V}_d \cup p$
21:             end while
22:         end if
23:     end if
24:     $H \leftarrow H \cup i \in \mathcal{V}_u$
25:     $\widehat{x}(i,t) = x(i,t)\ \forall\ i \in \mathcal{H}$
26: end for
27: return $\mathbf{x}, \widehat{\mathbf{x}}$
```

In each round, each sector s passes a message to its out-neighbors. That is, messages are passed along the edges E, between sectors. In the first round, this message contains the incoming vehicles into s, $\mathcal{V}_s^-$. In subsequent rounds, each sector appends $\mathcal{V}_s^-$ to each message that they received in the previous round and passes it along. A sector s knows that one of its incoming vehicles $v \in \mathcal{V}_s^-$ is part of a cycle if it sees v in a received message. Consider the example cycle shown in green in Fig. 2. The vehicles are labeled, with v2 currently in f2 and wanting to proceed into g2. In the first round, sector g2 sends the identity (but not position) of v2 to sector g1 and receives the identity of v1 from sector f2. In the next round, sector g2 sends a message with v2 and v1 to sector g1. Once sector g2 receives a message containing v2, it knows that v2 is part of a cycle and can prioritize it. While sector g2 can determine the identity of all the vehicles in the cycle, it cannot extrapolate the precise location of each vehicle for cycles longer than four vehicles. Moreover, it cannot identify the position of any vehicle in incoming messages that is not part of a cycle. This achieves the goal of finding cycles under our information sharing constraints.

D. Computing backpressure

We use backpressure to determine the order in which to deconflict sectors $\mathcal{S}$. To motivate why this is necessary, consider what would happen if sector d2 was deconflicted first, followed by e3 in Fig. 2. Sector d2 may choose to allow the vehicle from d1 to enter. This would block vehicles in

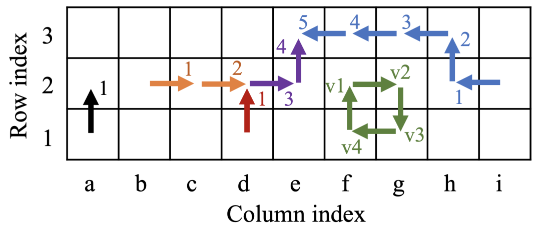


Figure 2: Example of system state with cycle in green (with vehicle IDs labeled) and non-cycle vehicles in other colors (with integers representing the backpressure).

b2 and c2, and force the vehicles in d2 and e2 to proceed. Note that the need to force vehicles out of currently occupied sectors would add additional communication overhead. When sector e3 is deconflicted, it would not get to choose between prioritizing vehicles in e2 and f3, because the vehicle in e2 must proceed to avoid gridlock from sector d2's deconfliction. This would lead to a suboptimal solution, as at most 3 non-cycle vehicles (occupying d1, d2, and e2) would be allowed to proceed, compared to possibly 5 non-cycle vehicles (occupying f3, g3, h3, h2, and i2). It would in fact be optimal to deconflict sector e3 first, followed by sector d2. Computing backpressure will allow us to do that.

In road networks, backpressure for a traffic movement can be the number of vehicles in a queue [13]. Queues with large build-ups are generally prioritized. We adapt similar logic to determine the order in which sectors are deconflicted. As with cycle detection, each sector s passes a message to its out-neighbors, $\mathcal{N}_s^+$. There are no rounds of messages, however. The message is a modified backpressure metric equal to the maximum number of vehicles that could proceed as a direct consequence of allowing v to proceed to out-neighbor r where $x(i,t) = s$ and $\widehat{x}(i,t) = r$. For example, in Fig. 2, sector e2 sends a backpressure value of 4 to e3 because at most 4 vehicles could proceed if the vehicle at e2 is permitted to enter e3. Note that sectors that are part of cycles do not pass backpressure messages. To compute backpressure, we start with sectors that do not have any in-neighbors but have some out-neighbors (e.g., a1, b2, d1, and i2). They pass a backpressure value of 1 to out-neighbors. Once a sector has received backpressure values from all of its in-neighbors, it sends the $\max(b_q) + 1,\ \forall q \in \mathcal{N}_s^-$ to all sectors $r \in \mathcal{N}_s^+$. For example, sector d2 sends a backpressure value of $\max(2,1)+1 = 3$ to sector e2. This process continues until all sectors with in-neighbors have received a backpressure value. With this heuristic, sector e3 would be deconflicted before d2, which gives the opportunity for the highest number of vehicles to proceed.

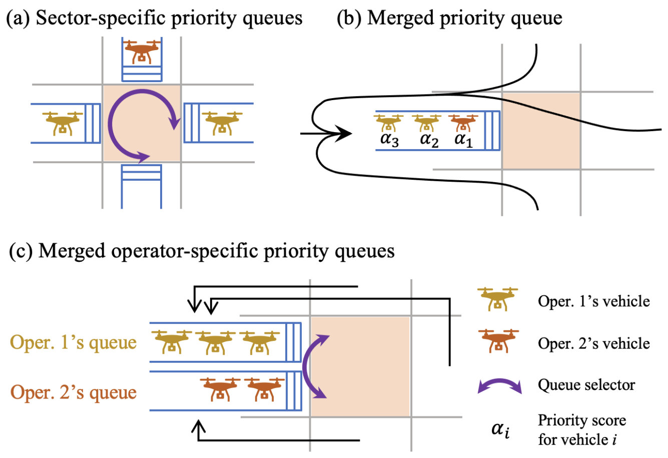
E. Baseline prioritization schemes

The most flexible and modular component of our algorithm is the PRIORITIZEVEHICLE function. The key characteristics of a sector-prioritization protocol is deciding the number of queues (e.g., one for each adjacent sector or one for each operator), the prioritization scheme used within a queue (e.g., time spent in queue or current delay of vehicle), and the logic for selecting a queue in case of multi-queue architectures (e.g., round-robin or random). Three candidate queue network architectures are presented in Fig. 3. Thus, a PRIORITIZEVEHICLE function requires that we specify the queue architecture (either (a), (b), or (c)), the queue selection logic, and the priority scheme for each queue.

We first present two intuitive baselines, which will be used for benchmarking the performance of our other schemes. Note that some of the schemes can be implemented differently depending on whether we are focused on vehicle-level efficiency and fairness or operator-level efficiency and fairness.

Random prioritization: We create a merged priority queue and assign a random priority score (α_i in Fig. 3(b)) to each of the vehicles.

Round-Robin prioritization: We use a sector-specific priority queue with first-come-first-served prioritization (i.e., highest priority for the vehicle which has been in the queue the longest). The queues are selected in a round-robin fashion with one vehicle allowed per queue per round. Round-robin prioritization can also be performed on an operator-level, whereby operator-specific priority queues are selected in a round-robin fashion as shown in Fig. 3(c).



(a) Sector-specific priority queues
(b) Merged priority queue
α_3 α_2 α_1
(c) Merged operator-specific priority queues
Oper. 1's queue
Oper. 2's queue
Oper. 1's vehicle
Oper. 2's vehicle
Queue selector
α_i
Priority score for vehicle i

Figure 3: Potential queue prioritization schemes for implementing the PRIORITIZEVEHICLE function.

IV. PRIORITIZATION SCHEMES

In this section we present four implementations of the PRIORITIZEVEHICLE function. The impact of these algorithms on efficiency and fairness will be discussed in Section V.

Backpressure prioritization: The backpressure metric is computed for each vehicle. When considering system-level performance, we use a merged priority queue, with the priority score equal to the backpressure for each vehicle. Backpressure prioritization can also be performed on an operator-level. In that case, we use a merged operator-specific queue. The queue corresponding to the operator with the highest total backpressure across all of its vehicles is chosen. Intuitively, the idea of this prioritization is to prioritize vehicles that have a higher potential to clear upstream congestion.

Backpressure prioritization can be shown to minimize the total system delays at that time-step (i.e., to minimize

$\sum_i del(i,t)$) among all possible solutions at time t. We do not include this proof here for brevity.

Accrued delay prioritization: Accrued delay is the delay that each vehicle has accumulated up to that point [26]. It can include delay accumulated during the current trip, as well as delay accumulated in previous trips operated by the same vehicle. Accrued delay prioritization orders vehicles based on their accrued delay, in descending order. More formally, this approach uses a merged priority queue, where the priority score for vehicle i, $\alpha_i = (\text{Accrued delay})_i$. The goal is to minimize additional delay for vehicles that have already been delayed. Accrued delay prioritization can be applied at the vehicle-level as described above, or at the operator level. In this case, a merged operator-specific priority queue architecture is employed, with the queue corresponding to the operator i with higher $\sum_{j\in\mathcal{V}_i}\alpha_j$ being prioritized.

Reversals prioritization: In this approach, a fair solution is one in which the relative ordering of arrivals at any resource is preserved according to the unimpeded schedule. Each vehicle keeps track of how many times it has encountered a resource in which its relative ordering was not preserved; this is called a reversal. For example, if vehicle A was originally scheduled to arrive at a resource before vehicle B, but instead vehicle B arrives before A, we count this as one reversal for A (and zero for B, since it benefited). Each vehicle keeps track of how many reversals it has experienced so far along its trajectory, so reversals may not necessarily have occurred at the current sector. Now, we set the priority score for vehicle i, $\alpha_i = (\text{reversals})_i$. For flight-specific prioritization, we use a merged priority queue, and when implementing an operator-specific algorithm, we use merged operator-specific priority queues.

Dominant resource fairness prioritization: Dominant Resource Fairness (DRF) is a solution for fair resource allocation with several desirable properties, like information sharing incentives and strategy-proofness [27]. For each operator and at each sector, we compute the resource share $R_o(s,t)$ that has been allocated to Operator o at sector s up through time t. When computing resource share, we account for resource allocation in the last T_b time-steps, with $T_b = 10$ assumed. If we define $U_o(s,t)$ as the capacity allocated to operator o in sector s at time t, resource share $R_o(s,t) = \sum_{i\in[t-T_b,t]}\frac{U_o(s,i)}{C(s,i)}$. The *dominant share* of an operator is the largest proportion of a resource that it consumes among all of the resources it uses ($D_o(t) = max(R_o(s,t)\ \forall s \in \mathcal{S})$. The dominant resource is the resource corresponding to the dominant share. The assumption with DRF is that equalizing dominant share among operators is fair. Thus, with each resource allocation, DRF prioritizes the operator with the lowest dominant share. In this paper, we consider DRF prioritization on an operator-level (not on a vehicle-level). We set the priority score for operator o to be $\alpha_o = 1/D_o(t)$, and use a merged operator-specific priority queue.

V. Numerical Evaluation of Protocols

We evaluate our protocol using three traffic scenarios: a) random flight patterns, b) cross-flows, and c) hub-based operations. We also discuss the impacts of intra-operator deconfliction before the protocol is run. Finally, we present the impacts on efficiency and fairness of flight-level prioritization and operator-level prioritization.

A. Scenario descriptions

Random flight trajectories (Scenario A): We generate a random scenario with two operators with 62 flights each departing across 50 time-steps, indexed by t. We define 13×13 square-shaped enroute "high" sectors, and the same number of sectors at ground-level ("low" sectors) to represent vertiports. Each low sector is assigned a random relative weight $\in (0,1)$. The sum of all relative weights is equal to 1. The origin and destination of each vehicle is randomly chosen based on the relative weights, simulating the fact that certain sectors will be more popular origins/destinations than others. Each vehicle's desired trajectory is assumed to be the shortest path trajectory from its origin to destination. Further, we assume that each vehicle transitions from the low sector to the high sector immediately after departure and from the high sector to the low sector upon arrival. We randomly sample the departure time based on a bi-modal distribution with the highest peak at $t = 40$ and another peak at $t = 20$. Based on the shortest path trajectory, we identify the sectors that each flight must cross. We assume a travel time greater than or equal to the number of sectors a vehicle traverses. A visualization of the traffic pattern is shown in the left side of Fig. 4(a). The ticks on the edge of the figure denote the size of the sector.

Cross-flow flight trajectories (Scenario B): We generate a scenario with two crossing flows. Operator 1 has 60 flights traveling in the east-west direction, whereas Operator 2 has 40 flights traveling north-south. Each operator has five possible origins and five possible destinations to achieve appropriately-oriented flows. Assumptions on sector geometry, departure time, and travel time remain the same as in Scenario A. This scenario helps evaluate our the performance of our protocol in unbalanced, cross-flow traffic. A visualization of the corresponding traffic pattern is shown on the left side of Fig. 4(b).

Hub-based trajectories (Scenario C): We use a hub-based package delivery scenario where four operators have warehouses on the outskirts of the city and make deliveries in locations randomly distributed around the city [28]. The demand is generated using a Poisson process. Operator 1 has hubs in the North and West with 66 flights, and Operator 2 has hubs in the South and East with 58 flights. A visualization of the traffic pattern is shown in the left side of Fig. 4(c).

In all three scenarios, each vehicle is required to transmit its current sector and its desired next sector to its next sector. If queried, vehicles also share fairness metrics, like accrued delay. We assume that each vehicle does not share other information, conforms to its route, and complies with the protocol, holding or proceeding as dictated. Recall that there are six prioritization schemes that we evaluate. Three of the six prioritizations (round robin, accrued delay, reversals) can be applied on a flight-level or operator-level; random and backpressure are only applied on a flight-level; and dominant-resource fairness (DRF) is only applied on an operator-level.

We conduct 100 trials for each prioritization scheme to account for the inherent randomness in the protocols. If all conflicting vehicles either a) belong to the same operator, or b) have the same priority according to the prioritization scheme (e.g., same accrued delay value), then backpressure is used as the tie-breaker for prioritization. We default to backpressure when fairness considerations are equal, since backpressure is expected to result in the minimum delay. However, if the fairness metric *and* backpressure of conflicting vehicles are the same, the sector randomly chooses to prioritize one of the vehicles. Such random tie-breaking usually happens in early time-steps and can have far-reaching and unpredictable downstream impacts on fairness and efficiency. We therefore conduct multiple trials for any given prioritization scheme and report the average values in the figures.

B. Intra-operator deconfliction

We assume that each operator has the full desired 4-D trajectory (three spatial dimensions plus time) of its vehicles before the start of the simulation. An operator may choose to deconflict its flights before the protocol ("intra-operator deconfliction". We use an unmanned traffic management flow (UTFM) optimization formulation to model the intra-operator deconfliction. We solve this model assuming the same capacity constraints, but without knowledge of other operators' flights. UTFM outputs a modified trajectory for each vehicle, and minimizes system total delay cost (TDC). TDC is the weighted sum of ground delay cost and airborne delay cost, where ground delay is preferred. Ground delays result in later departure times for vehicles, while airborne delays lead to longer occupancy times in sectors. The details of the optimization formulation to solve this strategic problem can be found in [18]. The original trajectory of these vehicles is still used for computing delay and fairness metrics.

With intra-operator deconfliction by Operator 1, if there were no other operators, there would be no conflicts (and no need for the protocol). With the addition of Operator 2, there will be conflicts, but fewer than there would have been without intra-operator deconfliction, since vehicles of Operator 1 will have fewer conflicts among their trajectories. Conflicts between vehicles of Operator 1 may still occur with *intra-operator* deconfliction, due to delays resulting from *inter-operator* conflicts. That is, even though Operator 1 deconflicted its vehicles in advance, further delays will be required to deconflict with vehicles of other operators, putting them in conflict with their own vehicles that they were initially deconflicted with.

We compared the results of each prioritization scheme with/without intra-operator deconfliction. We present the results on random flight trajectories scenario (Scenario A) with flight-level prioritization. We assume that both Operator 1 and Operator 2 independently conduct intra-operator deconfliction ("with" case), or neither of them do ("without" case). With intra-operator deconfliction, the operators added a total of 47 minutes of delay prior to the start of the simulation. Table I shows the percentage change in a) total delay, b) standard deviation of flight delay, and c) number of conflicts when applying intra-operator deconfliction. In every prioritization mechanism tested, applying intra-operator deconfliction reduced the number of conflicts by around 30%, but increased total delay and standard deviation by 6-12%. This implies that the delay saved due to the reduced number of conflicts did not make up for the initial 47 minutes of delay applied. Flight accrued delay is particularly affected because intra-operator deconfliction increases accrued delay of some flights before the simulation. Then, the accrued delay prioritization expedites these delayed flights at the expense of others.

TABLE I. Percentage change with intra-operator deconfliction relative to without intra-operator deconfliction for Scenario A.

Prioritization	Total Delay	Std. Deviation	Conflicts
Round Robin	8.8%	8.2%	-29.8%
Accrued Delay	11.8%	7.8%	-29.5%
Reversals	6.3%	10.5%	-31.0%
DRF	6.1%	11.8%	-27.7%
Back-pressure	11.8%	12.8%	-25.7%

It may be counter-intuitive that intra-operator deconfliction deteriorates the final solution. However, intra-operator deconfliction is inefficient in this case because the operator does not have sufficient information about other operators to make informed decisions. These simulations do not represent the approaches to strategic deconfliction proposed in the literature that include discovery and synchronization of intent information from other operators [7]. Instead, in these simulations operators do not have any incentive to share flight information to other operators because of competitive or privacy concerns. Thus, it is better for operators to send their vehicles out at their originally scheduled times and rely on the deconfliction protocol. In optimization problems, it may be better to solve a global problem rather than several sub-problems. This is the case for these protocols as well.

C. Fairness and efficiency of flight-level prioritizations

We first discuss the results of flight-level prioritization, shown in the second column of Fig.4. From this point on, we assume no intra-operator deconfliction, as we have shown how it reduces system efficiency and fairness. We use mean standard deviation of flight delay (henceforth referred to as "standard deviation") as a metric for fairness, and mean total system delay ("total delay") as a metric for efficiency. (Recall that we show the mean because we ran 100 trials for each prioritization scheme.) We show the results of five prioritization schemes. Note that DRF is not shown in the center panels of Fig. 4 because we only consider DRF on an operator-level.

Some initial observations are that Scenario B with cross-flow trajectories has by far the highest delay and standard deviation across all prioritizations, even though it has the fewest number of flights. This is because it has the highest number of conflicts, particularly in the center. Scenario C with hub-based trajectories has the next highest total delay and standard deviation, because it has more conflicts than Scenario A.

We now consider the best and worst performers in terms of fairness and efficiency. There is a pattern consistent across all scenarios in terms of relative efficiency and fairness

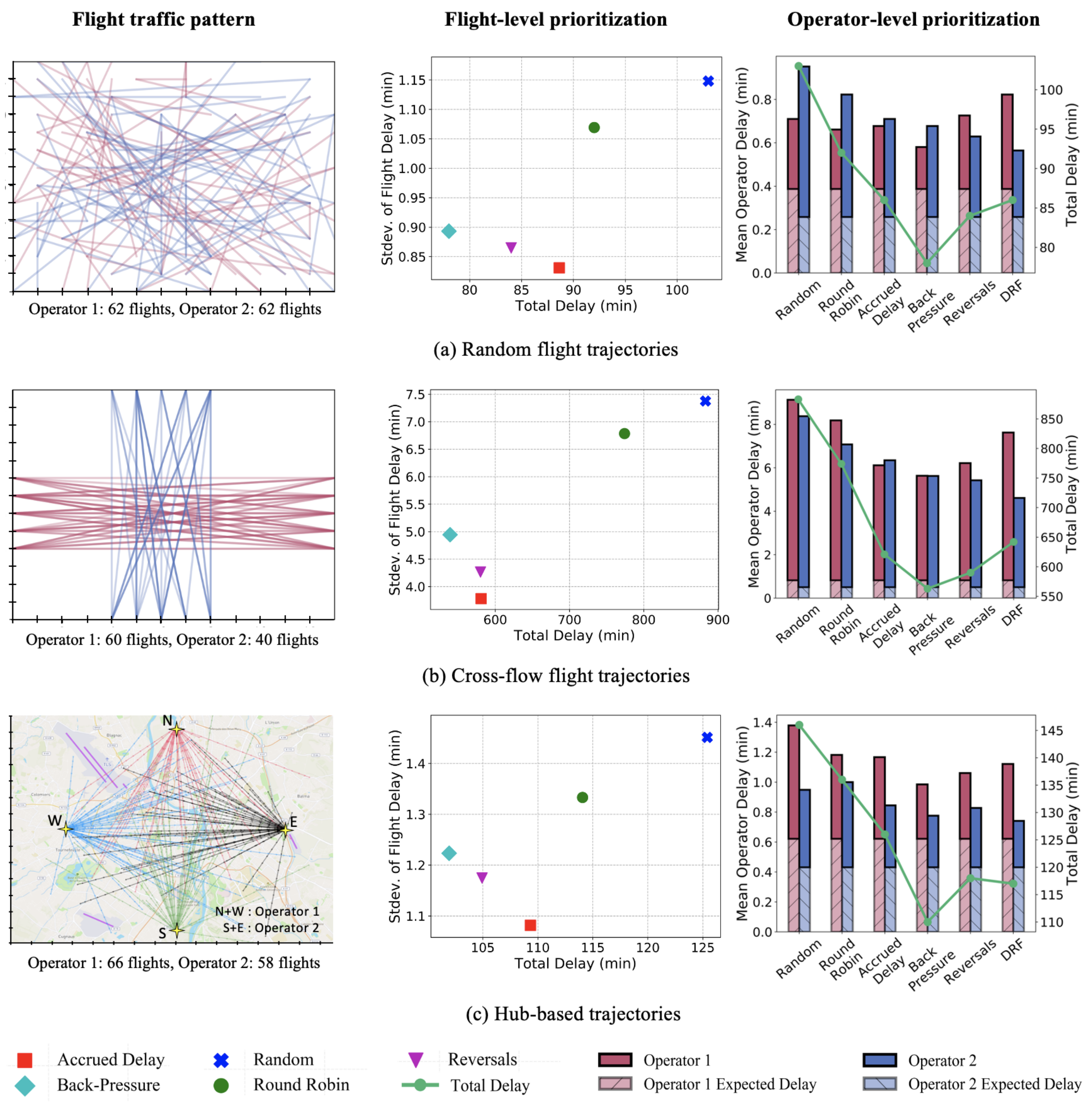


Figure 4: Efficiency and fairness of our flight-level and operator-level protocols for three traffic scenarios.

ordering between the prioritization schemes. Prioritizing by accrued delay (red square) leads to the lowest standard deviation across all scenarios. This makes sense given that this approach can balance final delay values by holding flights with little accrued delay and prioritizing flights with high accrued delay. Back-pressure (teal diamond) has the lowest total delay across all scenarios, and therefore highest efficiency, which is expected given its proven properties in other settings. Prioritizing randomly is the least fair and least efficient outcome, which matches our expectations. Round-robin prioritization leads to more fair and efficient solutions than random prioritization, but still worse than the other four prioritizations. Reversals prioritization (purple triangle) is in between backpressure and accrued delay in terms of total delay and standard deviation. We hypothesize that there is a Pareto frontier of different prioritizations that lie between the solutions of backpressure prioritization and accrued delay prioritization, and reversals is one of them.

D. *Fairness and efficiency of operator-level prioritizations*

We now consider the performance of operator-level prioritization, shown in the third column of Fig. 4. We show the results of six prioritization schemes. Note that random and backpressure are shown for reference even though they are prioritized on the flight-level. As with flight-level prioritization, we evaluate efficiency with total delay, shown in green on the right-hand side axis. The trends in total delay with operator-level fairness are similar to those with flight-level fairness. Backpressure prioritization remains the most efficient, whereas random and round-robin prioritizations are the least efficient. Reversals, DRF, and accrued delay prioritizations have similar efficiency, with accrued delay having the highest total delay.

For evaluating operator-level fairness, we do not show the standard deviation of flight delays, since prioritizations occur on an operator level. We instead show mean operator delay and mean expected delay, which we use to present three possible notions of operator fairness below:

1) **Equal mean operator delay:** The idea is to equalize μ_o^{oper} across operators $o \in \mathcal{O}$.
2) **Equal mean excess operator delay:** The idea here is that equal mean delays for operators might not capture the fact that one operator might have an original schedule that has significantly more inherent delays than the other. For example, in scenario (b), clearly Operator 1 has more flights that cause congestion, even in the absence of Operator 2. We define the mean *expected delay* as the average delay experienced by an operator, if it were the only user of the system. Denote by μ_o^{exp} the mean expected delay of operator o. We define the mean *excess delay* as $\mu_o^{exc} = \mu_o^{oper} - \mu_o^{exp}$. Excess delay is philosophically similar to the concept of time-order deviation introduced in [29], but for multiple flights and operators instead of multiple congested resources. Thus, one notion of fairness could be equalizing the excess delay across operators.
3) **Target excess delay ratio:** We may want to link the excess operator delay ratio with the expected operator delay ratio. Suppose $\frac{\mu_1^{exp}}{\mu_2^{exp}} > 1$. For each protocol and scenario, we can construct the relation between excess delay ratio and expected delay ratio as $\frac{\mu_1^{exc}}{\mu_2^{exc}} = \alpha \frac{\mu_1^{exp}}{\mu_2^{exp}}$. We can choose values of α to target. A value of $\alpha < 1$ implies that the protocol deemphasizes imbalances in the expected delays when assigning excess delays. By contrast, $\alpha = 1$ means that excess delays were assigned in proportion to the expected delays, and $\alpha > 1$ implies that the imbalances in the expected delays were further exacerbated when assigning excess delays.

The fairness of prioritization schemes depends on the metric used. For example, with backpressure prioritization with the cross-flow scenario shown in Fig. 4(b), Operator 1 and Operator 2 have nearly identical mean delays, which satisfies operator fairness notion 1. On the other hand, Operator 2 may feel that this is unfair given that Operator 1 has 50% more flights and 63% more expected delay. Thus, Operator 2 may prefer fairness notions 2 or 3. Note that in all scenarios, Operator 2 has lower mean expected delay than Operator 1 (and in Scenarios B and C, fewer flights). Because DRF tries to equalize resource allocation between operators, in all scenarios, Operator 2 receives the lowest delay with DRF.

Fig. 5 shows the three notions of operator fairness across the six prioritizations and the three scenarios. Backpressure generally has the most equal mean operator delay and mean excess operator delay. The next best two prioritizations in terms of these two notions of operator fairness are accrued delay then reversals. DRF has disparate mean operator and mean excess operator delays (particularly in Scenario B), but it has α values closest to 1. This indicates that DRF could be appealing in situations where excess delays should be assigned in proportion to expected delays. Note that in Scenario C, DRF has $\alpha = 1.12 > 1$, indicating that the excess delay ratio exceeded the expected delay ratio.

There is an inherent trade-off between efficiency (total delay) and operator fairness. For example, backpressure prioritization leads to the lowest delays but also low α values, which may be unfair depending on the target. At the other extreme, DRF has α values close to 1, but higher total delays. On both counts, accrued delay and reversals fall in-between backpressure and DRF.

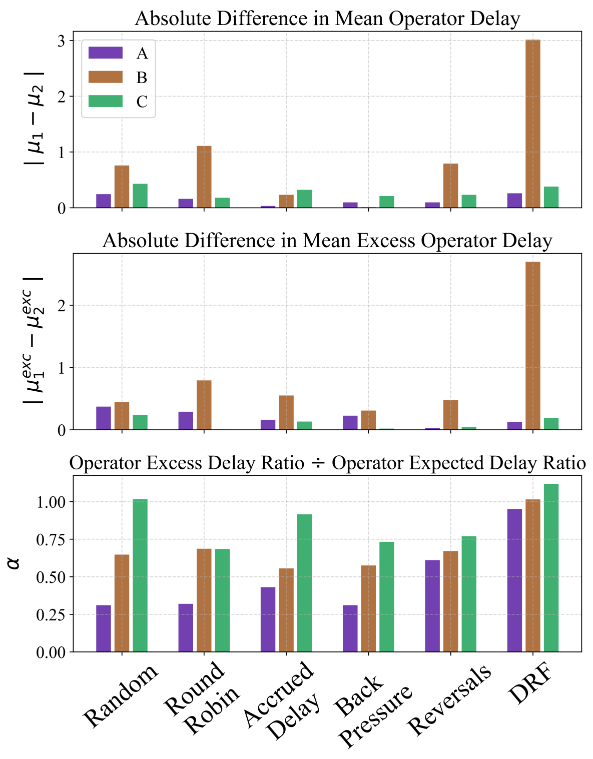


Figure 5: Operator notions of fairness across prioritizations for Scenarios A-C. The notion of operator fairness is indicated on top of each plot and on the y-axis label.

VI. Conclusions

This paper explored the concept of reduced-information and fair congestion management algorithms for efficient advanced air mobility operations. Such algorithms are critical to the development of a traffic management infrastructure that can support dynamic, low lead-time operations such as drone deliveries and urban air mobility. Our key contribution lies in developing reduced-information decentralized protocols that avoid gridlocks, a frequent pitfall of approaches that do not rely on centralized coordination. Our protocol is also flexible and supports a wide variety of user-specified priorities to help achieve the desired balance between efficiency and fairness at an operator- or system-wide level.

Our work is the first step in understanding the interplay between information exchange, efficiency, and fairness in traffic coordination. In this regard, our analysis is also of relevance to the problem of coordination and control of road traffic, especially with the increasing deployment of connected autonomous vehicles. We believe that there are several directions for future work. First, our analysis in this paper is largely simulation-based, and there is significant scope for deriving theoretical guarantees on their performance. For example, we have been able to prove that the backpressure algorithm is indeed the minimal myopic (one-step) delay solution. Operators and flights can easily manipulate the protocols by misreporting their current state or strategically filing for particular routes or at specific times to minimize their overall delays. Identifying robust, strategy-proof, and incentive-compatible congestion management protocols is therefore of practical interest. Finally, adding more realism by incorporating reroutes and simultaneously handling both on-demand and scheduled aircraft operations will improve the practical application of the proposed approaches.

References

[1] National Academies of Sciences, Engineering, and Medicine, *Advancing Aerial Mobility: A National Blueprint*. Washington, DC: The National Academies Press, 2020. [Online]. Available: https://www.nap.edu/catalog/25646/advancing-aerial-mobility-a-national-blueprint

[2] A. Majumdar, W. Y. Ochieng, J. Bentham, and M. Richards, "En-route sector capacity estimation methodologies: An international survey," *Journal of Air Transport Management*, vol. 11, no. 6, pp. 375–387, 2005.

[3] J. Rios, I. Smith, P. Venkatesan, J. Homola, M. Johnson, and J. Jung, "UAS Service Supplier Specification: Baseline requirements for providing USS services within the UAS Traffic Management System," NASA, Tech. Rep., 2019.

[4] K. Balakrishnan, J. Polastre, J. Mooberry, R. Golding, and P. Sachs, "Blueprint for the Sky: The roadmap for the safe integration of autonomous aircraft," Airbus UTM, Tech. Rep., 2018.

[5] Federal Aviation Administration, "Urban Air Mobility: Concept of Operations v1.0," FAA, Tech. Rep., 2020.

[6] D. Bertsimas and S. S. Patterson, "The air traffic flow management problem with enroute capacities," *Operations research*, vol. 46, no. 3, pp. 406–422, 1998.

[7] P. Kopardekar, J. Rios, T. Prevot, M. Johnson, J. Jung, and J. E. Robinson, "Unmanned aircraft system traffic management (utm) concept of operations," in *AIAA aviation forum*, 2016.

[8] S. H. Low, F. Paganini, and J. C. Doyle, "Internet congestion control," *IEEE control systems magazine*, vol. 22, no. 1, pp. 28–43, 2002.

[9] A. Atta, S. Abbas, M. A. Khan, G. Ahmed, and U. Farooq, "An adaptive approach: Smart traffic congestion control system," *Journal of King Saud University-Computer and Information Sciences*, 2018.

[10] H. Khadilkar and H. Balakrishnan, "Network congestion control of airport surface operations," *Journal of Guidance, Control, and Dynamics*, vol. 37, no. 3, pp. 933–940, 2014.

[11] A. Eryilmaz and R. Srikant, "Fair resource allocation in wireless networks using queue-length-based scheduling and congestion control," *IEEE/ACM transactions on networking*, vol. 15, no. 6, pp. 1333–1344, 2007.

[12] S. Badrinath, M. Z. Li, and H. Balakrishnan, "Integrated surface–airspace model of airport departures," *Journal of Guidance, Control, and Dynamics*, vol. 42, no. 5, pp. 1049–1063, 2019.

[13] J. Gregoire, X. Qian, E. Frazzoli, A. De La Fortelle, and T. Wongpiromsarn, "Capacity-aware backpressure traffic signal control," *IEEE Transactions on Control of Network Systems*, vol. 2, no. 2, pp. 164–173, 2014.

[14] X. Sun and Y. Yin, "A simulation study on max pressure control of signalized intersections," *Transportation research record*, vol. 2672, no. 18, pp. 117–127, 2018.

[15] M. W. Levin and D. Rey, "Conflict-point formulation of intersection control for autonomous vehicles," *Transportation Research Part C: Emerging Technologies*, vol. 85, pp. 528–547, 2017.

[16] S. Lu, V. Bharghavan, and R. Srikant, "Fair scheduling in wireless packet networks," *IEEE/ACM Transactions on networking*, vol. 7, no. 4, pp. 473–489, 1999.

[17] A. D. Evans, M. Egorov, and S. Munn, "Fairness in Decentralized Strategic Deconfliction in UTM," in *AIAA Scitech 2020 Forum*, 2020, p. 2203.

[18] C. Chin, K. Gopalakrishnan, M. Egorov, A. Evans, and H. Balakrishnan, "Efficiency and fairness in unmanned air traffic flow management," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–13, 2021.

[19] D. Bertsimas and S. Gupta, "Fairness and collaboration in network air traffic flow management: an optimization approach," *Transportation Science*, vol. 50, no. 1, pp. 57–76, 2015.

[20] M. Ribeiro, J. Ellerbroek, and J. Hoekstra, "Review of conflict resolution methods for manned and unmanned aviation," *Aerospace*, vol. 7, no. 6, p. 79, 2020.

[21] E. D'Amato, M. Mattei, and I. Notaro, "Distributed reactive model predictive control for collision avoidance of unmanned aerial vehicles in civil airspace," *Journal of Intelligent & Robotic Systems*, vol. 97, no. 1, pp. 185–203, 2020.

[22] I. Hwang, J. Kim, and C. Tomlin, "Protocol-based conflict resolution for air traffic control," *Air Traffic Control Quarterly*, vol. 15, no. 1, pp. 1–34, 2007.

[23] H. Y. Ong and M. J. Kochenderfer, "Markov decision process-based distributed conflict resolution for drone air traffic management," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 1, pp. 69–80, 2017.

[24] E. Cruck and J. Lygeros, "A mathematical framework for subliminal air traffic control," in *AIAA Guidance, Navigation and Control Conference and Exhibit*, 2007, p. 6693.

[25] R. Rocha and B. Thatte, "Distributed cycle detection in large-scale sparse graphs," in *Proceedings of the Simposio Brasileiro de Pesquisa Operacional Pernambuco, Brazil*. Sobrapo, 2015, pp. 25–28.

[26] H. Idris, C. Chin, and A. D. Evans, "Accrued delay application in trajectory-based operations," in *USA/Europe Air Traffic Management R&D Seminar*, 2019.

[27] A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker, and I. Stoica, "Dominant resource fairness: Fair allocation of multiple resource types." in *Nsdi*, vol. 11, no. 2011, 2011, pp. 24–24.

[28] M. Egorov, V. Kuroda, and P. Sachs, "Encounter aware flight planning in the unmanned airspace," *2019 Integrated Communications, Navigation and Surveillance Conference (ICNS)*, 2019.

[29] C. Barnhart, D. Bertsimas, C. Caramanis, and D. Fearing, "Equitable and efficient coordination in traffic flow management," *Transportation science*, vol. 46, no. 2, pp. 262–280, 2012.

Author Biographies

Christopher Chin is a PhD Candidate in the Department of Aeronautics and Astronautics at MIT. His research interests include optimization and protocols in air traffic management, as well as crew/airline scheduling.

Karthik Gopalakrishnan is a PhD Candidate in the Department of Aeronautics and Astronautics at MIT. He is interested in the modeling, optimization, and control of air transportation networks.

Maxim Egorov is a research scientist at Airbus UTM where he works on research and development of novel software technologies for the next generation of air traffic management.

Antony Evans is the Traffic Management System Architect at Airbus UTM where he works on the system design of digital traffic management solutions for the next generation of aircraft and operations.

Hamsa Balakrishnan is the William E. Leonhard (1940) Professor of Aeronautics and Astronautics at MIT. Her research interests are in the design, analysis, and implementation of control and optimization algorithms for large-scale cyber-physical infrastructures, with an emphasis on air transportation systems.